The Report Committee for Jae Hong Min

Certifies that this is the approved version of the following report:

Low-Power Fused FFT Butterfly Arithmetic Unit With Merged Multiple-Constant Multiplier

APPROVED BY

SUPERVISING COMMITTEE:

Supervisor:

Earl E. Swartzlander, Jr.

Mircea D. Driga

Low-Power Fused FFT Butterfly Arithmetic Unit With Merged Multiple-Constant Multiplier

by

Jae Hong Min, B.S., B.E.

Report

Presented to the Faculty of the Graduate School

of the University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2010

For my family and Soo-Hyun Yang

Acknowledgements

Above all, I would like to thank our Father in heaven for having mercy and love on me. I am grateful to my supervisor, Professor Earl E. Swartzlander, Jr. for his guidance and support on my master's report as well as for giving me the opportunity to pursue my Ph.D degree at The University of Texas at Austin. I also deeply thank professor Mircea D. Driga for sparing his time to read my master's report and for his generous feedback.

I greatly appreciate my family for their sincere and grateful support. I also would like to thank my girl friend, Soo-Hyun Yang and her parents for their encouragement on my study. Last but not least, I would like to thank my colleagues, Junyoung Park and Jongho Lee who took various courses with me here at UT. The time of discussion and collaboration that I spent with them was truly invaluable.

December 2010

Abstract

Low-Power Fused FFT Butterfly Arithmetic Unit With Merged Multiple-Constant Multiplier

by

Jae Hong Min, M.S.E

The University of Texas at Austin, 2010

SUPERVISOR: Earl E. Swartzlander, Jr.

Fused floating-point arithmetic units such as a floating-point fused Dot-Product (fused DP) and a floating-point fused Add-Subtract (fused AS) are employed for the implementation of the butterfly unit of the FFT due to their characteristics of low power and less area. In addition, the fused DP has less delay and lower error. Among the elements of the fused DP, two internal mantissa multipliers occupy the largest area and consume the largest power. A Multiple-Constant Multiplier (MCM) architecture has high speed, low power consumption, and small area compared to a conventional multiplier. The MCM is used for the internal mantissa multiplier, providing a solution for low power and high performance. Despite the benefits of the MCM, it lacks precision compared to a conventional multiplier. Due to this, the butterfly unit using the MCM has higher error.

In this report, a new architecture of the butterfly unit has been designed by merging conventional MCMs. The new architecture provides two options. It either reduces the error or it lowers the power compared to a conventional MCM butterfly unit.

Table of Contents

Page

Acknowledgements v

Abstract vi

Table of Contents vii

List of Tables ix

List of Figures x

Chapter 1 – Introduction 1

1.1 Motivation 1

1.2 Overview of IEEE-754 1

1.3 Floating-Point Adder 3

1.4 Floating-Point Multiplication 4

Chapter 2 - Approach for Low-Power Fast Fourier Transform 6

2.1 Multiplier-less Architecture 6

2.2 Floating-Point Fused Butterfly Architecture 8

2.2.1 Floating-Point Fused Multiply-Add 8

2.2.2 Floating-Point Fused Add-Subtract 10

2.2.3 Comparison of Approaches for Radix-2 Butterfly Unit 12

Chapter 3 - High-Level Modeling for Floating-Point Arithmetic 14

3.1 High-Level Modeling Plan 14

3.2 Input Conversion Error 15

3.3 Conventional Floating-Point Adder 15

3.4 Conventional Floating-Point Multiplication 16

3.5 Conventional Two-Term Dot-Product Unit 17
3.6 Floating-Point Fused Dot-Product Unit 18
3.7 Comparison between Fused and Conventional Dot-Product Unit 19
Chapter 4 - High-Level Modeling for Butterfly Unit 22
Chapter 5 - Butterfly Unit with Merged MCMs 28
5.1 Butterfly Unit with MCM 28
5.2 New Butterfly Unit implemented by merging Neighbor MCMs 31
Chapter 6 - Conclusion and Future Work 35
Bibliography 36
VITA 38

List of Tables

Table 1.1: IEEE-754 Single Precision and Double Precision Format. 2
Table 1.2: Range of Single Precision and Double Precision Floating-Point Numbers [16]. 2
Table 1.3: Special Values of the IEEE Floating-Point System [16]. 3
Table 2.1: Two-Term Dot-Product Unit Delay Comparison [5]. 9
Table 2.2: Two-Term Dot-Product Unit Area Comparison [5]. 10
Table 2.3: Two-Term Dot-Product Unit Power Consumption Comparison [5]. 10
Table 2.4: Add-Subtract Unit Delay Comparison. 11
Table 2.5: Add-Subtract Unit Area Comparison. 11
Table 2.6: Add-Subtract Unit Power Consumption Comparison 12
Table 2.7: Comparison of Conventional and Fused Butterfly Unit [14] 13
Table 3.1: Average Relative Absolute Error of Conventional and Fused DP 20
Table 3.2: Variance of Relative Error of Conventional and Fused DP 20

List of Figures

Figure 2.1: Floating-Point Fused Dot-Product Unit. [10]. 9
Figure 2.2: Floating-Point Fused Add-Subtract Unit. [10]. 11
Figure 2.3: Fused Add-Subtract Unit and Fused Dot Product Unit Concept [14]. 12
Figure 2.4: FFT Radix-2 Butterfly Computation [14]. 13
Figure 3.1: High-Level Modeling Plan. 14
Figure 3.2: Histogram Representing Relative Error from Conversion. 15
Figure 3.3: Histogram Representing Relative Error of a Conventional Floating-Point Adder. 16
Figure 3.4: Histogram Representing Relative Error of a Conventional Floating-Point Multiplier. 17
Figure 3.5: Conventional Parallel Dot-Product Unit. 17
Figure 3.6: Error Model of the Conventional Floating-Point Dot-Product Unit 18
Figure 3.7: Error Model of the Floating-Point Fused Dot-Product Unit 19
Figure 3.8 Histogram of Relative Error for the Four Dot-Product Models 20
Figure 3.9 Average Relative Absolute Error for the Four Dot-Product Models 21
Figure 3.10 Variance of Relative Error the Four Dot-Product Models 21
Figure 4.1: Radix-2 Decimation in Frequency Butterfly. 22
Figure 4.2: Conventional Radix-2 DIF Butterfly Computation [14]. 22
Figure 4.3: Fused Radix-2 DIF Butterfly Computation [14]. 23
Figure 4.4: Average Relative Absolute Error of X_{re} and X_{im} 24
Figure 4.5: Average Relative Absolute Error of Y_{re} and Y_{im} 25
Figure 4.6: Variance of Relative Error of X_{re} and X_{im} 26
Figure 4.7: Variance of Relative Error of Y_{re} and Y_{im} 27
Figure 5.1: Average Relative Absolute Error of MCM 29
Figure 5.2: Variance of Relative Error of MCM 29

Figure 5.3: Block Diagram of a Conventional Multiple Constant Multiplier in Butterfly Unit 30
Figure 5.4: Percentage of the Same Shift-Coefficient 31
Figure 5.5: Merged MCM 32
Figure 5.6: Butterfly Unit with Merged MCM 32
Figure 5.7: Decrease of Average Relative Absolute Error (%) 33
Figure 5.8: Average Relative Absolute Error with Different Types of Butterfly Unit . 34

Chapter 1 – Introduction

1.1 Motivation

Recently, merging general purposed CPU and graphic processors using floating-point arithmetic into one silicon die has become a way to achieve high performance. With this trend, a design for a low-power floating-point arithmetic unit is important. If the floating-point unit consumes much power, the system can fail because the CPU and the floating-point unit are in the same die using the same power supplier. This report focuses on the low-power design for a FFT butterfly unit because the FFT is one of the most important signal processing application.

1.2 Overview of IEEE-754

The floating-point number system is attractive for various applications such as graphics, signal processing, and scientific calculations because it can provide an enormous dynamic range. This large range of the floating-point number system enables designers to overcome problems such as overflow that often occur in fixed point implementation. Among the various floating-point number formats that are used, the IEEE-754 standard is used in this report [1]. The IEEE-754 floating point standard is frequently used in Intel, Apple, and most Unix systems [3].

A floating-point number consists of a sign bit, exponent bits, and mantissa bits. The exponent is biased in order to allow the small as well as large numbers [16]. The following equation shows how the floating-point number, N can be described by the sign bit, exponent bits, and mantissa bits. The mantissa is called the significand in the IEEE standard.

$$N= (-1)^{sign} * 2^{(exponent-bias)} *mantissa \quad (1)$$

The size of the exponent bits affects the range of the floating-point numbers. The size of the mantissa influences the precision of the floating-point number. In the normal mode, all the mantissas are of the form 1.xxxx⋯; the leading 1 is hidden in the format. Table 1.1 shows the size of the exponent and the mantissa in single precision (SP) and double precision (DP) formats.

Table 1.1: IEEE-754 Single Precision and Double Precision Format.

Precision	bits	Sign bit	Exponent bits	Mantissa bits
SP	32	1	8	23
DP	64	1	11	52

DP has higher resolution than SP due to the larger mantissa. Moreover, DP has a larger range than SP because of the larger exponent; Table 1.2 demonstrates the range of SP and DP.

Table 1.2: Range of Single Precision and Double Precision Floating-Point Numbers [16].

Precision	Bias	Exp_{min}	Exp_{max}	N_{min}	N_{max}
SP	127	1	254	$2^{(1-127)}*1.00\ldots0$	$2^{(254-127)}*1.11\ldots1$
DP	1023	1	2046	$2^{(1-1023)*}1.00\ldots0$	$2^{(2046-1023)}*1.00\ldots0$

Floating-point numbers between ±Nmin and ±Nmax are normal values. Floating-point numbers other than the normal value are special values. When the floating-point number is between zero and $\pm N_{min}$, the number is called a denormal number. The resolution of the denormal number is constant between $-N_{min}$ and $+N_{min}$. Table 1.3 shows special values of the floating-point system.

Table 1.3: Special Values of the IEEE Floating-Point System [16].

Exponent	Mantissa	Value
00..0	00…0	Zero
00..0	0.00…1 to 0.11…1	Denormal
11…1	1.00…0	Infinity
11…1	1.0…01 to 1.011..1	SNaN
11…1	1.1…01 to 1.11…1	QNaN

1.3 Floating-Point Adder

A floating-point adder is a basic block but its algorithm is more complicated than a floating point multiplier. The following equation is the result of floating point addition: $Z(S_z, E_z, M_z) = X(S_x, E_x, M_x) + Y(S_y, E_y, M_y)$. The output Z follows the IEEE-754 format.

Algorithm 1.1 shows how the floating-point adder operates. For high-speed implementation of floating-point adder, several modifications have been developed. One is the double-path implementation. The normalization requires many left shifts when the operation is subtraction and when the output has many leading zeros. Therefore, two types of data path-CLOSE and FAR- are used for high performance floating point addition [2].

Algorithm 1.1 Floating-Point Adder.

START:

$D = E_x - E_y$

E_z = Max (E_x , E_y)

ALIGNMENT:

if $D > 0$ **then**

Shift right the significand M_y *by* D positions

else

Shift right the significand M_x *by* D positions

end if

(S_z, M_z) = Signed ADD {(S_x, M_x), (S_y, M_y)}

NORMALIZATION:

if M_z is overflowed **then**

Shift right M_z by one

Increase E_z by one

else if M_z has leading zeros

Shift left M_z by the number of leading zeros

Decrease E_z by the number of lading zeros

else No action

end if

ROUND:

Round(M_z)

If M_z is overflow **then**

Shift right M_z by one

Increase E_z by one

end if

Determine exception flags and special values

1.4 Floating-Point Multiplication

The following equation represents the processing of floating-point multiplication: $Z(S_z, E_z, M_z) = X(S_x, E_x, M_x) * Y(S_y, E_y, M_y)$. Algorithm 2 shows operation of the floating-point multiplication.

Algorithm 1.2 Floating-Point Multiplication.

START:

$E_z = E_x + E_y - Bias$

$M_z = M_x * M_y$

$S_z = S_x$ xor S_y

NORMALIZATION:

if M_z is overflowed **then**

Shift right M_z by one

Increase E_z by one

else No action

end if

ROUND:

Round(M_z)

If M_z is overflow **then**

Shift right M_z by one

Increase E_z by one

end if

Determine exception flags and special values

The output from the mantissa multiplication doubles the mantissa size. This is decreased by a rounding operation. Through the rounding operation, error is generated. The floating point multiplication algorithm is simpler than floating point addition algorithm. However, the overall area of floating-point multiplier is larger than the area of floating-point adder [6].

Chapter 2 - Approach for Low-Power Fast Fourier Transform

An advanced Fast Fourier Transform (FFT) architecture and an improved butterfly architecture have been suggested to produce a low-power FFT. The butterfly unit is the most power consuming portion of the FFT. Designing a low-power butterfly unit is the key technique for a low-power FFT. In this section, a low-power architecture for a butterfly unit is discussed.

2.1 Multiplier-less Architecture

Reducing the amount of computation is the key technique to reduce the power consumption. In the butterfly unit, it is well known that the total number of complex multiplications is reduced by removing trivial complex multiplications, such as multiplications by 0, 1 and -1. Furthermore, the constant coefficients of FFT play an important role in employing a multiplier-less architecture. For a multiplier-less architecture, the Multiple Constant Multiplier (MCM) replaces the normal multiplier; since the normal multiplier is one of the most power-consuming elements in the butterfly arithmetic unit. While the normal multiplier performs a large number of additions, the MCM consists of series of several shifters and adders. This reduces the complexity, power, and size of the non-trivial multipliers [9]. For efficient MCM operation, Canonical Signed-Digit (CSD) and 2's complement numbers are used together since their combined usage minimizes the number of addition/shift operations. The following example shows the multiplier-less algorithm.

Example 2.1 Twiddle Factor Computation For Non-Trivial Factors [9].

Pre-computing: $5*X = X + (X << 2)$ and $65*X = X + (X << 6)$
Multiplication: $5a82*X = (5*X << 12) + (5*X << 9) + (65*X << 1)$
$7641*X = (X << 15) + (65*X << -5*X << 9)$
$30fb*X = (65*X << 8) - (X << 12) - 5*X$

The precision of the MCM's output is decreased when the number of MCM internal shifters is less than the required number of shifts for 0% error; however, fewer shifters reduce calculation delay and power consumption. Example 2.2 demonstrates how the accuracy decreases.

Example 2.2 Reduced Accuracy of MCM.

1 0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0 1 0 0 0 (Coefficient =*490F88*)
1-1 0 1-1 0 1-10 0 0 1 0 0 0 0-1 0 0 1-1 0 0 0 (Booth-recoded coefficient =*490F88*)
1 0 0 1 0 0 1 0 0 0 1 0 0 0 0-1 0 0 0 1 0 0 0 (Modified Booth-recoded coefficient = *490F88*)
If the number of shifters is four for MCM, the coefficient is
1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 (Modified Booth-recoded coefficient =*491000*)
The relative error is $2.5062*10^{-5}$.

Devices that use MCM or conventional multipliers have been compared in terms of power and area in several papers [9], [11], and [12]. The experiment implementing multiply-and-accumulate (MAC) with a conventional multiplier and with a three-shifter-based MCM shows that the MAC with MCMs saves 56% of the dynamic power, 20.2% of the leakage power, and 21.6% of the area compared to the MAC with conventional multipliers [12].

2.2 Floating-Point Fused Butterfly Architecture

Butterfly arithmetic of the FFT can employ the floating-point Fused Multiply-Add (FMA) unit and the floating-point Fused Add-Subtract (FAS) unit since these units have benefits on area, and power; additionally, FMA has lower delay and reduced error. The characteristics of both units are described in the following sub-section.

2.2.1 Floating-Point Fused Multiply-Add

In 1990, IBM developed a floating-point fused multiply-add (FMA) unit for RISC system 6000 [7], [8]. In the RISC system 6000, they merged a floating-point multiplier and a floating-point adder. The floating-point fused multiply-add unit performs X*Y without rounding and normalization and then immediately adds C. The FMA unit has been used for the butterfly arithmetic unit due to its higher performance, less error, lower power consumption, and smaller area compared to the conventional floating-point dot-product unit [10]; the FMA has been modified to compute two-term dot-products. The two-term dot-product is Y= A*B + C*D. Figure 2.1 presents the block diagram of the floating-point fused dot-product unit (Fused DP).

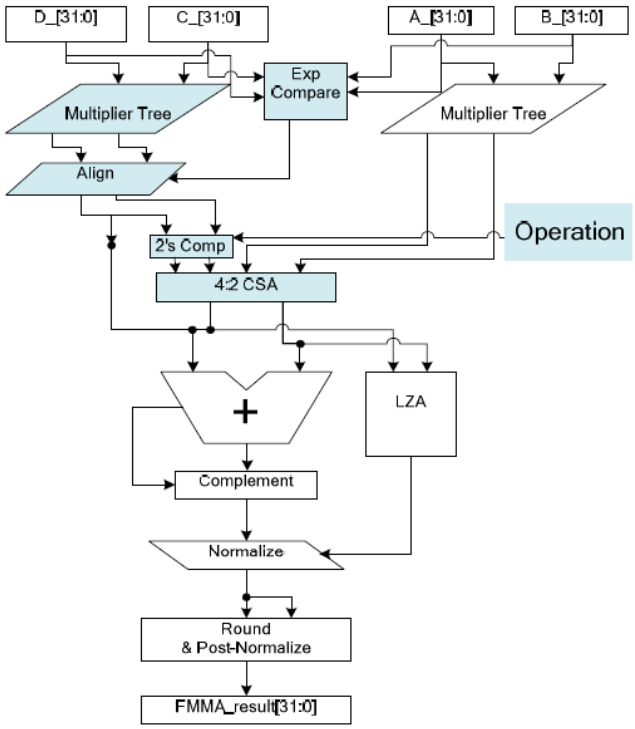


Figure 2.1: Floating-Point Fused Dot-Product Unit. [10].

Saleh compared several types of two-term dot-product units in terms of power, area, and delay [5]. The floating two-term dot-product unit was implemented using bulk-CMOS 45nm process. Tables 2.1, 2.2, and 2.3 show the delay, area, and power consumption of the conventional parallel floating-point Dot-Product (conventional DP) unit and the floating-point fused Dot-Product (fused DP) unit. Compared to the conventional DP unit, the fused DP unit is 16% less in delay, 33% less in area, and 20% less in power consumption.

Table 2.1: Two-Term Dot-Product Unit Delay Comparison [5].

Unit	Delay	% of Floating-Point Multiplier
Conventional parallel dot-product	3.23 ns	179
Fused dot-product	2.72 ns	151

Table 2.2: Two-Term Dot-Product Unit Area Comparison [5].

Unit	Area (μm^2)	% of Floating-Point Multiplier
Conventional parallel dot-product	24,043	254
Fused dot-product	16,104	170

Table 2.3: Two-Term Dot-Product Unit Power Consumption Comparison [5].

Unit	Avg' Power	% of Floating-Point Multiplier
Conventional parallel dot-product	42.39 mW	193
Fused dot-product	33.90 mW	154

2.2.2 Floating-Point Fused Add-Subtract

If both the sum and difference of the same pair of numbers is needed, the floating-point Fused Add-Subtract unit (FAS) consumes less area and power compared with the conventional parallel floating-point adder and subtractor because it shares a single exponent compare and shift with both the adder and the subtractor. Due to this, the FAS is preferred for the butterfly unit. Figure 2.2 is a block diagram of a FAS [10]. Adding green blocks to the conventional floating point adder and modifying the yellow blocks of the conventional floating point adder leads to the FAS.

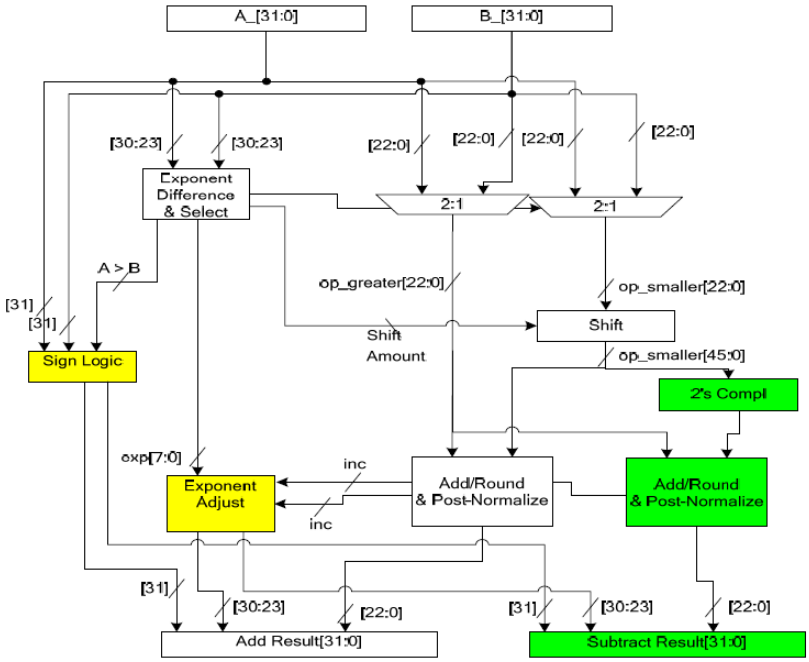


Figure 2.2: Floating-Point Fused Add-Subtract Unit. [10].

Tables 2.4, 2.5, and 2.6 show the comparison between the conventional parallel floating-point add-subtract and floating-point fused add-subtract unit [10] - using a bulk-CMOS 45nm process.

Table 2.4: Add-Subtract Unit Delay Comparison.

Unit	Delay	% of Floating-Point Adder
Conventional parallel add-subtract	1.70 ns	104
Fused add-subtract	1.72 ns	105

Table 2.5: Add-Subtract Unit Area Comparison.

Unit	Area (μm^2)	% of Floating-Point Adder
Conventional parallel add-subtract	7,456	196
Fused add-subtract	5,947	156

Table 2.6: Add-Subtract Unit Power Consumption Comparison.

Unit	Avg' Power	% of Floating-Point Adder
Conventional parallel add-subtract	12.83 mW	190
Fused add-subtract	10.15 mW	150

The FAS unit consumes 20% less area and power than the conventional parallel floating-point add-subtract unit while the delay remains similar.

2.2.3 Comparison of Approaches for Radix-2 Butterfly Unit

The fused DP and the fused AS unit described in the previous sections are used to construct a radix-2 butterfly unit; Figure 2.3 demonstrates symbols of fused DP and fused AS. Figure 2.4 shows the butterfly unit with floating point fused arithmetic [14].

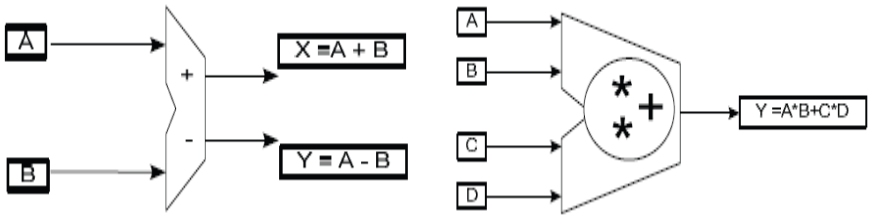


Figure 2.3: Fused Add-Subtract Unit and Fused Dot Product Unit Concept [14].

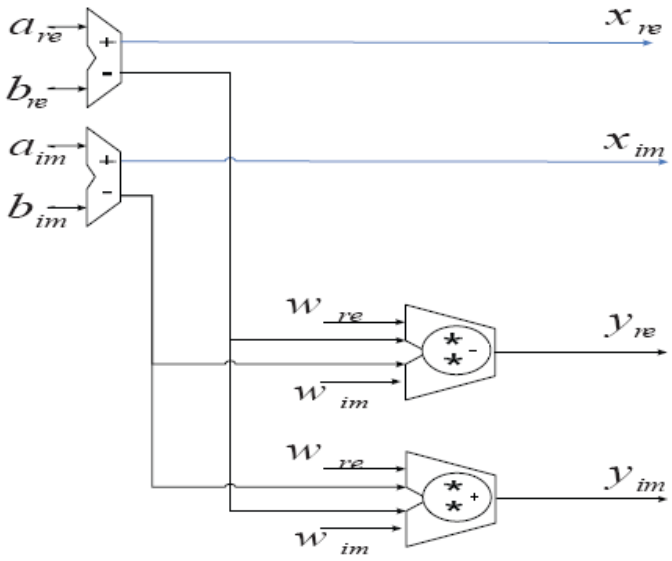


Figure 2.4: FFT Radix-2 Butterfly Computation [14].

The fused radix-2 butterfly unit has an advantage over the conventional radix-2 butterfly in terms of area, speed, and power consumption [14]. The following table shows the result of implementation with the bulk CMOS 45nm library.

Table 2.7: Comparison of Conventional and Fused Butterfly Unit [14].

	Conventional Butterfly	Fused Butterfly
Format	IEEE 754 Single-Precision	
Standard Cell Area	53,778 μm^2	38,626 μm^2
Critical Timing Path	4.6 ns	4.0 ns
Total Power	13.4 mW	12.1 mW

Chapter 3 - High-Level Modeling for Floating-Point Arithmetic

3.1 High-Level Modeling Plan

In this project, both a floating-point fused arithmetic and the multiply-less architecture were applied to design the low-power butterfly unit. First, high-level modeling and simulation were used to verify the function of the butterfly using the floating-point fused arithmetic and the multiply-less architecture. MathWorks MATLAB was used for the high-level modeling. After verifying and analyzing the high-level modeled butterfly unit, the butterfly unit was improved in power and precision; the solution for the improvement was merging neighbor MCMs. Figure 3.1 shows the detailed plan of this project.

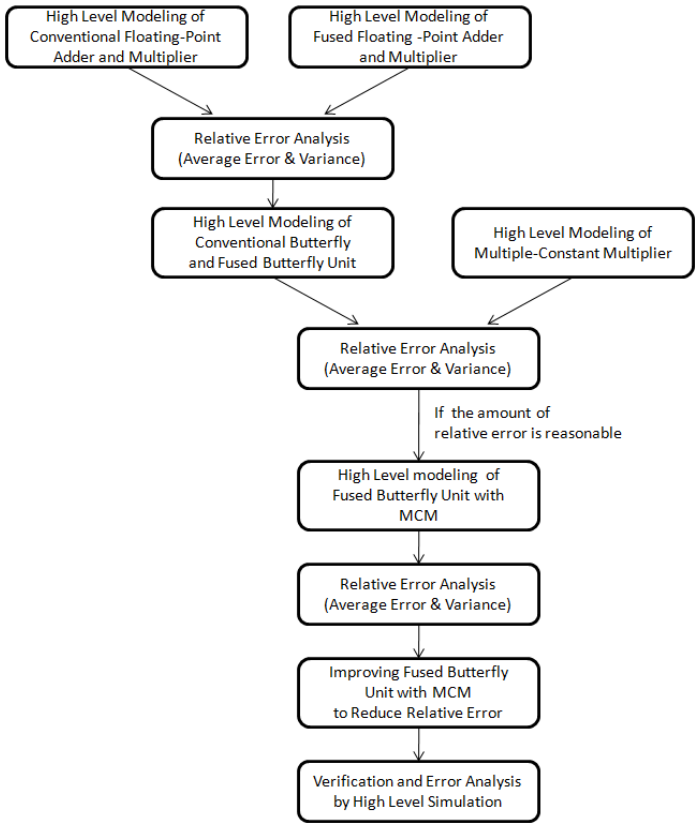


Figure 3.1: High-Level Modeling Plan.

3.2 Input Conversion Error

Converting the decimal value to the binary value causes error. Example 3.1 shows an example of the input conversion error. Figure 3.2 shows a histogram representing the relative error caused by the input conversion.

Example 3.1 Input Conversion Error.

Input: -0.001161239153570,
Converted Sign bit: 1
Converted Exponent: '1110101'
Converted Mantissa: '11000001101001011000'
Relative Error from Input Conversion : 3.7578 $*10^{-8}$

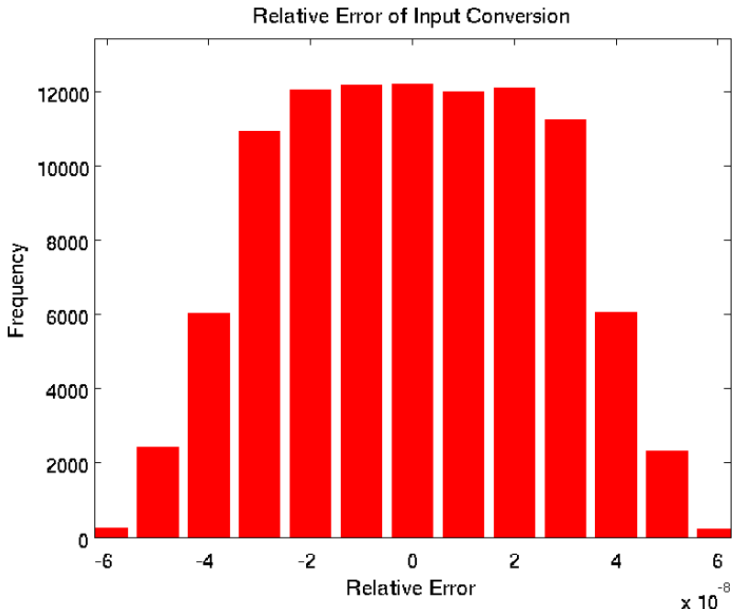


Figure 3.2: Histogram Representing Relative Error from Conversion.

3.3 Conventional Floating-Point Adder

First in this project, the conventional floating-point adder was built as a high-level model. This model followed Algorithm 1.1 as described in Chapter 1. The Round to

Nearest Even (RNE) rounding algorithm was used. The average relative absolute error of the conventional floating-point adder is $1.14808*10^{-8}$. The variance of the relative error is $3.2798*10^{-16}$. The following histogram shows the relative errors of the floating-point adder with a set of 50,000 inputs.

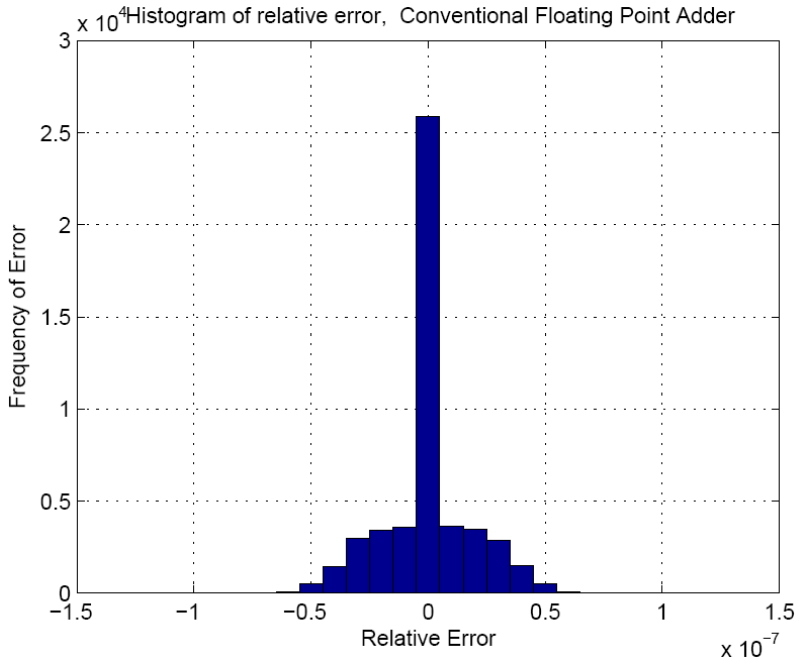


Figure 3.3: Histogram Representing Relative Error of a Conventional Floating-Point Adder.

3.4 Conventional Floating-Point Multiplication

This floating-point multiplier model followed Algorithm 1.2 as described in Chapter 1. Round to Nearest Even (RNE) rounding algorithm was used. The average relative absolute error of the conventional floating-point multiplication is $2.1470*10^{-8}$. The variance of the relative error is $6.3940*10^{-16}$. The following histogram shows the relative errors of the floating-point multiplier with a set of 50,000 inputs.

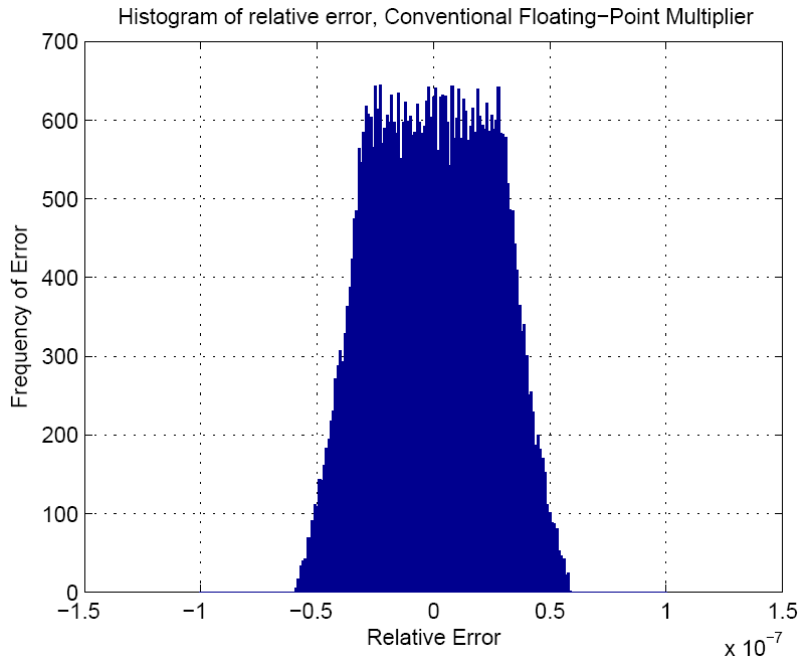


Figure 3.4: Histogram Representing Relative Error of a Conventional Floating-Point Multiplier.

3.5 Conventional Two-Term Dot-Product Unit

The conventional two-term Dot-Product (conventional DP) unit consists of two conventional multipliers and one conventional adder. Figure 3.5 shows a simple block diagram of the conventional DP unit.

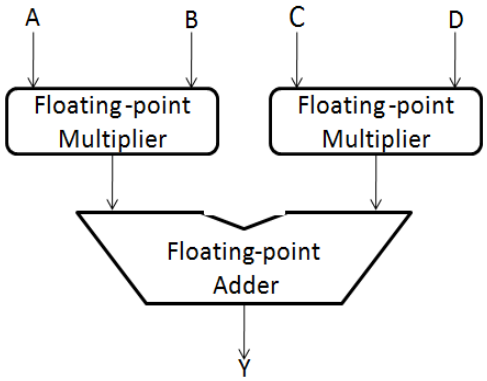


Figure 3.5: Conventional Parallel Dot-Product Unit.

The two multipliers of the conventional DP are operated and the outputs from the two multipliers are summed by the adder. The two multipliers and the adder round their outputs. There are three rounding operations (one after each multiplier and one after the adder). Figure 3.6 shows the error model of the conventional DP unit. The relative error analysis of the conventional dot-product unit will be described in Section 3.5 'Comparison of Conventional and Fused Dot-Product unit.'

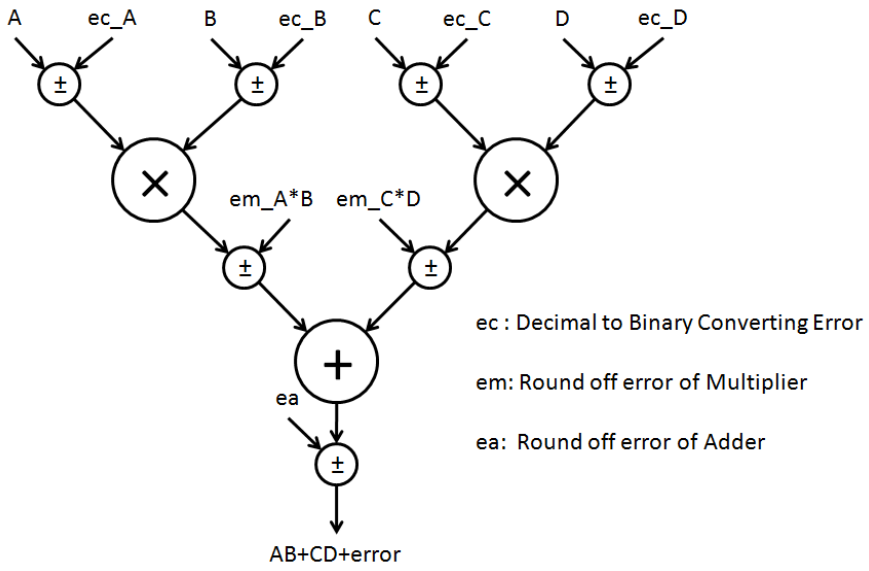


Figure 3.6: Error Model of the Conventional Floating-Point Dot-Product Unit.

3.6 Floating-Point Fused Dot-Product Unit

A floating-point fused Dot-Product (fused DP) unit was designed as a high-level model by extending Algorithm 3.1 for Fused Multiply-Add. The algorithm was modified for the fused two-term dot-product unit. Figure 3.7 presents the error model of the fused DP.

Algorithm 3.1 Floating-Point Fused Multiply-Add Unit.

START:

$E_{temp} = E_x + E_y$

$M_{temp} = M_x * M_y$

$S_{temp} = S_x$ xor S_y

Pre-shift M_c by $m+3$. ; m is bit-width, Pre-shift allows avoiding bidirectional shifts.

ALIGNMENT:

$D = E_x + E_y - E_c + m + 3 - Bias$

if $D > 0$ **then**

Shift right D positions the significand M_c

Maximum shift = $3m+1$

else

No action

end if

(S_z, M_z) = Signed ADD $\{(S_{temp}, M_{temp}), (S_c, M_c)\}$

Normalize S_z, M_z

Round S_z, M_z

Determine exception flags and special values

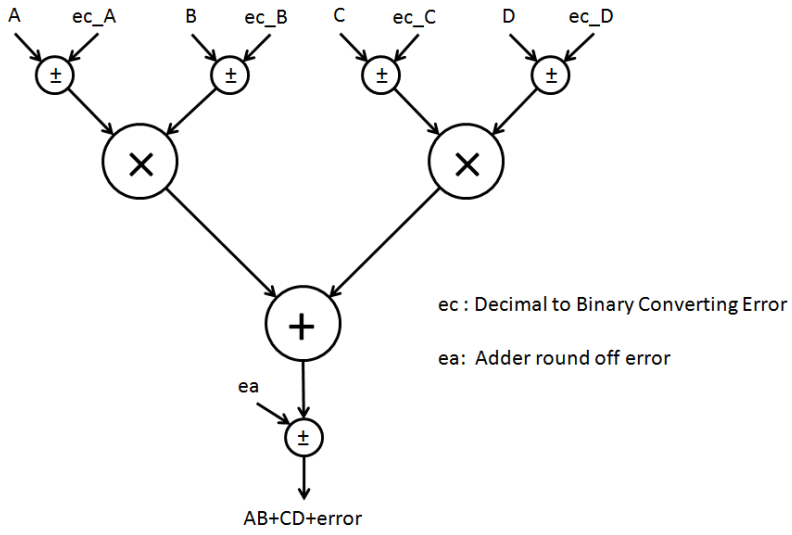


Figure 3.7: Error Model of the Floating-Point Fused Dot-Product Unit.

3.7 Comparison between Fused and Conventional Dot-Product Unit

The average and variance of the fused DP relative error is smaller than that of the conventional DP. Figure 3.8 shows the histogram of the relative error from four types of dot-product models: fused DP and conventional DP with/without input conversion error.

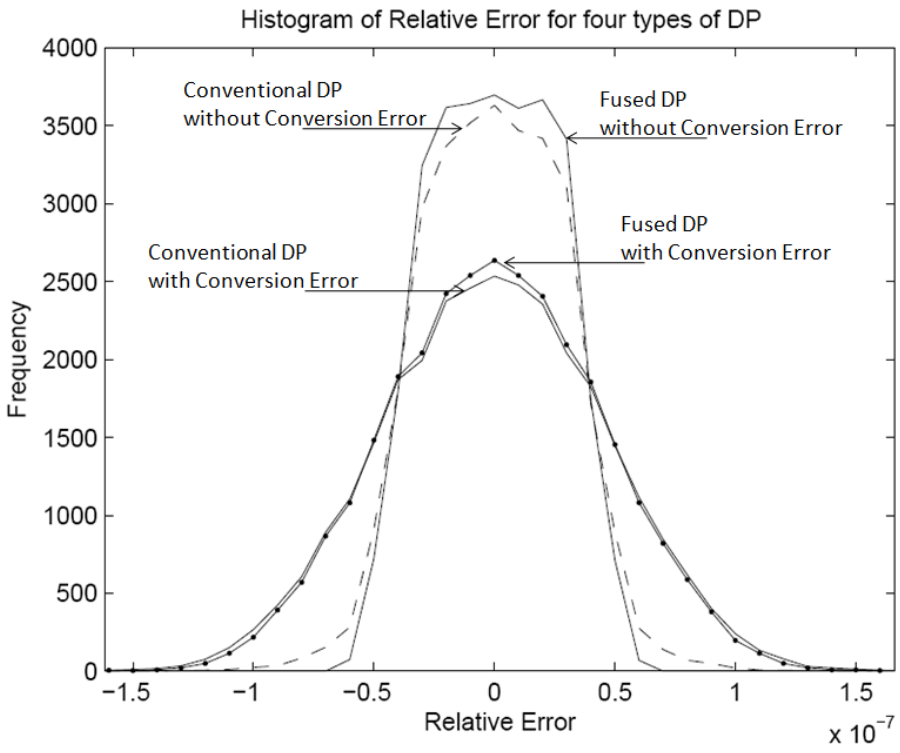


Figure 3.8 Histogram of Relative Error for the Four Dot-Product Models.

The following tables and figures show the average relative absolute error and the variance of relative error.

Table 3.1: Average Relative Absolute Error of Conventional and Fused DP.

Fused DP with Conversion Error	Conventional DP with Conversion Error	Fused DP without Conversion Error	Conventional DP without Conversion Error
$3.2548*10^{-8}$	$3.4486*10^{-8}$	$2.0694*10^{-8}$	$2.3324*10^{-8}$

Table 3.2: Variance of Relative Error of Conventional and Fused DP.

Fused DP with Conversion Error	Conventional DP with Conversion Error	Fused DP without Conversion Error	Conventional DP without Conversion Error
$2.4198*10^{-15}$	$2.6041*10^{-15}$	$6.5610*10^{-16}$	$8.7073*10^{-16}$

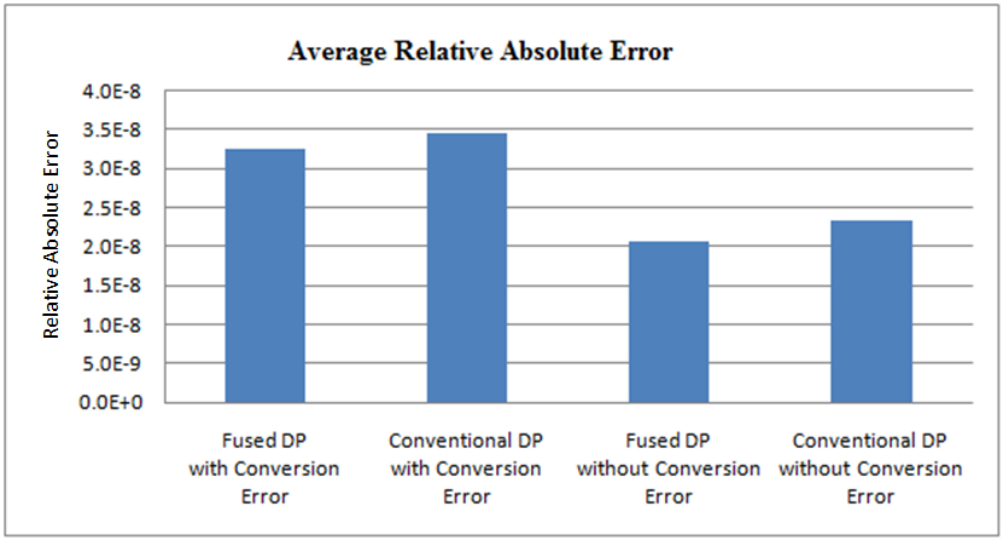


Figure 3.9 Average Relative Absolute Error for the Four Dot-Product Models.

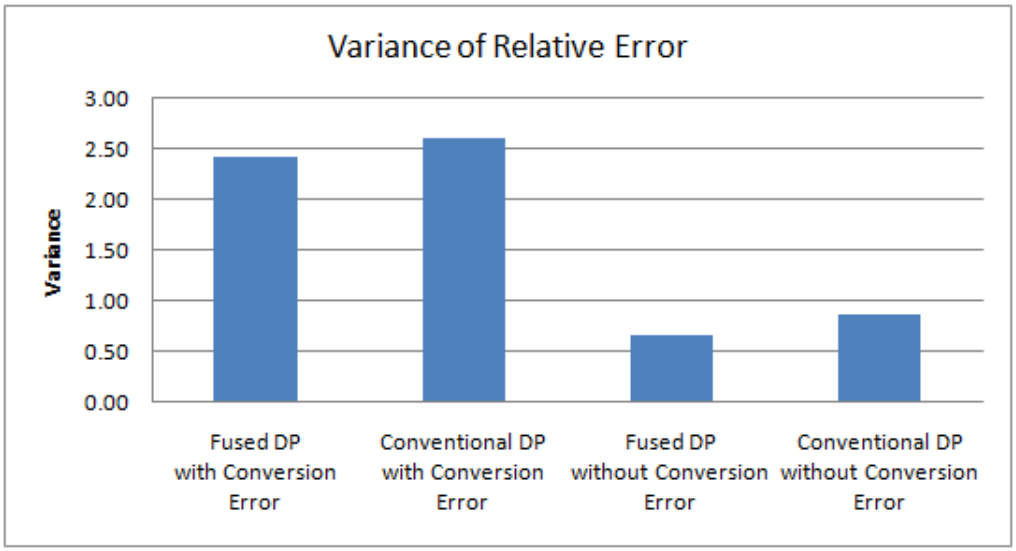


Figure 3.10 Variance of Relative Error for the Four Dot-Product Models.

Chapter 4 - High-Level Modeling for Butterfly Unit

This chapter introduces high-level models for the conventional and fused butterfly units. The relative errors of the conventional and fused butterfly units were analyzed. The fused butterfly unit has lower average relative absolute error and variance compared to the conventional butterfly unit. In this report, the conventional and fused Radix-2 butterfly unit was modeled for a 1024-point FFT. These units make use of complex multiplication. The following figure shows the radix-2 decimation in frequency (DIF) butterfly that was used. The variables a, b, w, x, and y are complex numbers. Figures 4.2 and 4.3 present block diagrams of a conventional butterfly unit and a fused butterfly unit.

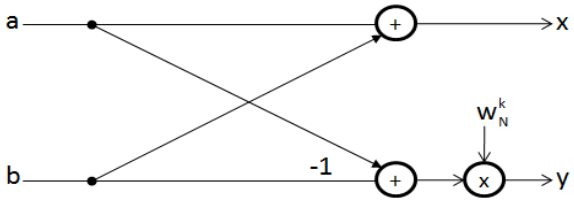


Figure 4.1: Radix-2 Decimation in Frequency Butterfly.

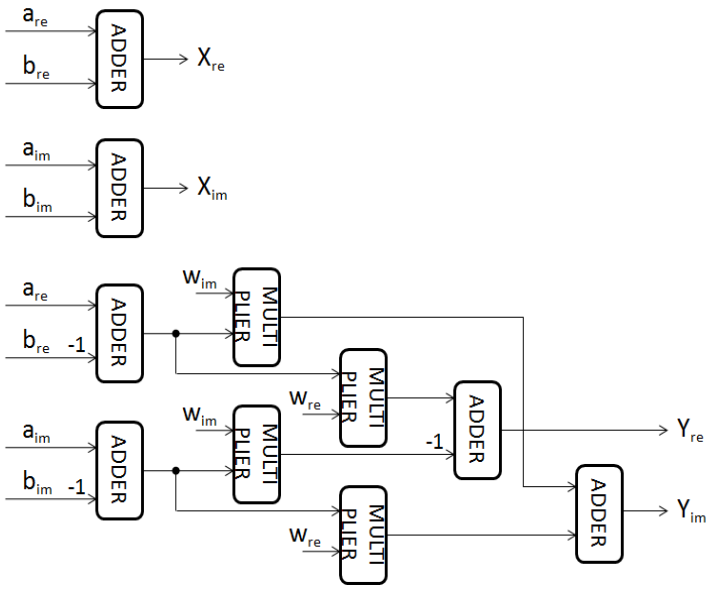


Figure 4.2: Conventional Radix-2 DIF Butterfly Computation [14].

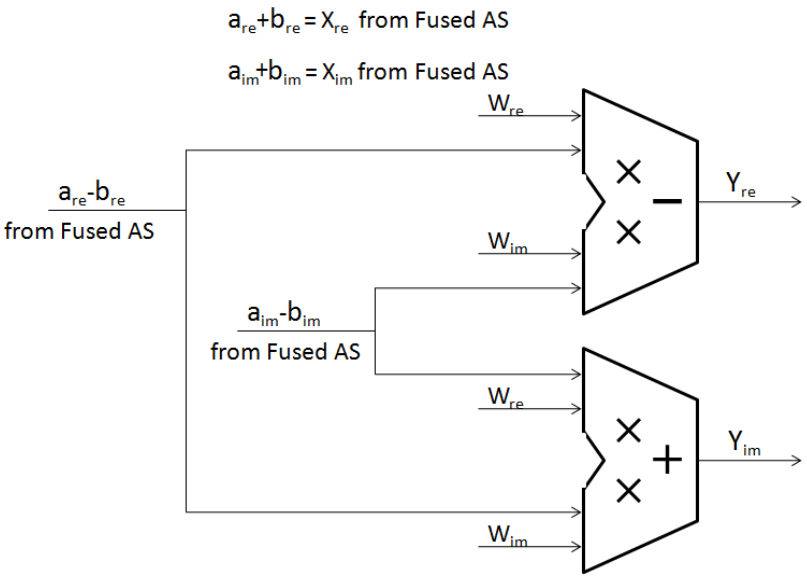
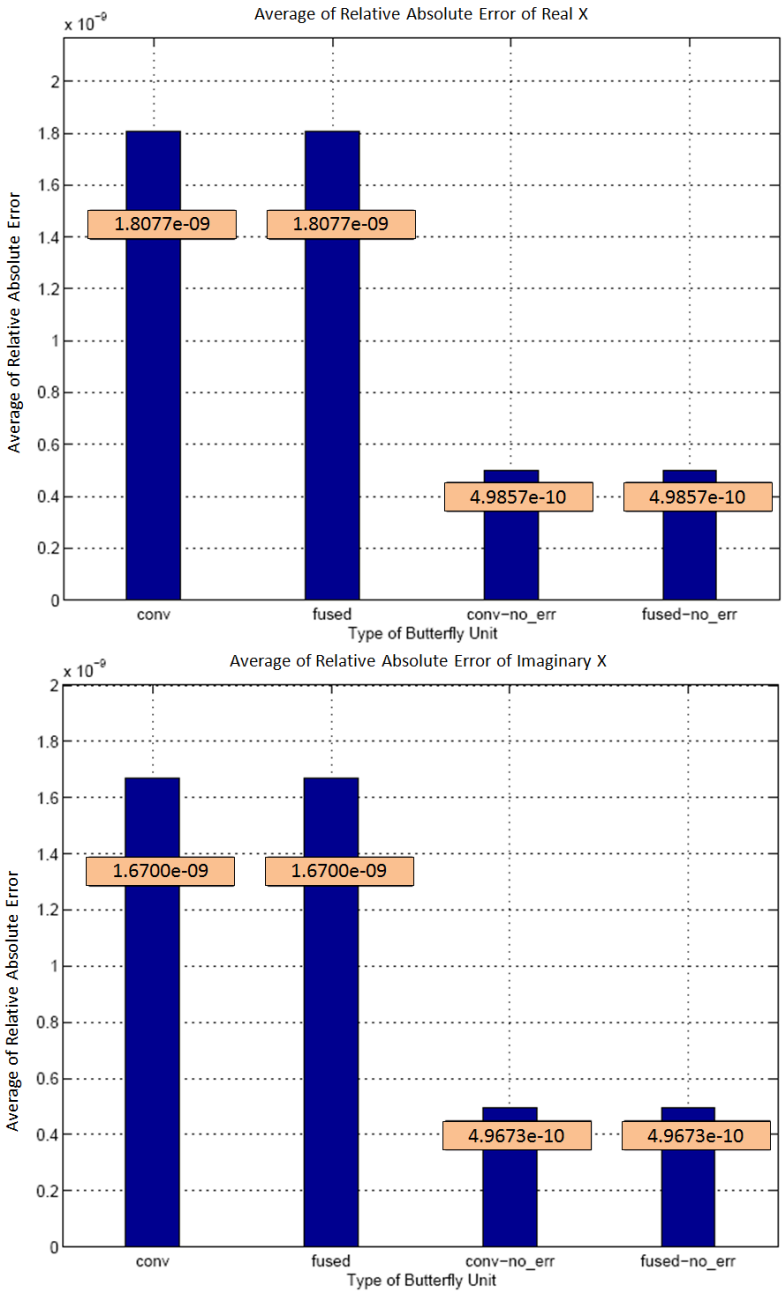


Figure 4.3: Fused Radix-2 DIF Butterfly Computation [14].

The relative errors of the conventional and fused butterfly unit were compared by simulation with a set of 3 million inputs. Figures 4.4 and 4.5 show the average relative absolute errors. Figures 4.6 and 4.7 present the variance of the relative error.

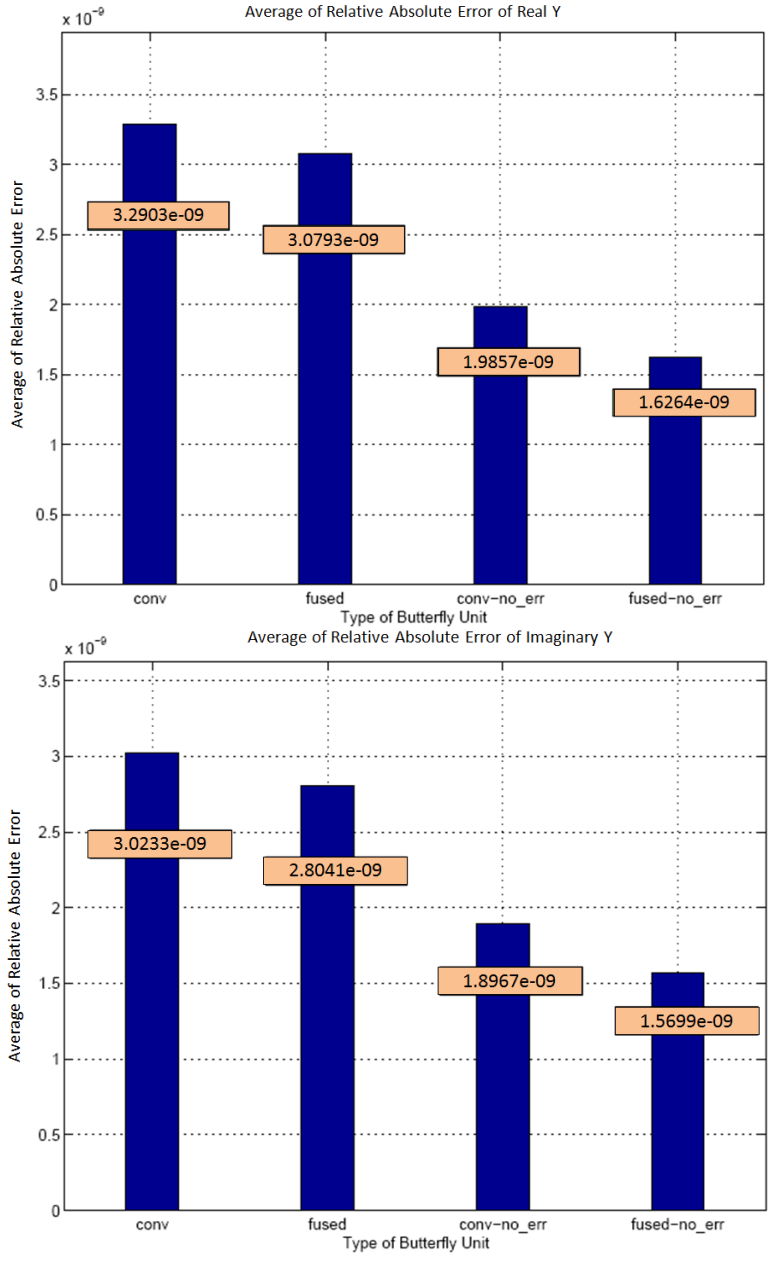


no_err means 'without input conversion error'

Figure 4.4: Average Relative Absolute Error of X_{re} and X_{im}.

The average relative absolute error of X_{re} and X_{im} from the fused butterfly unit is the same as that of the conventional butterfly unit because X_{re} and X_{im} of the fused

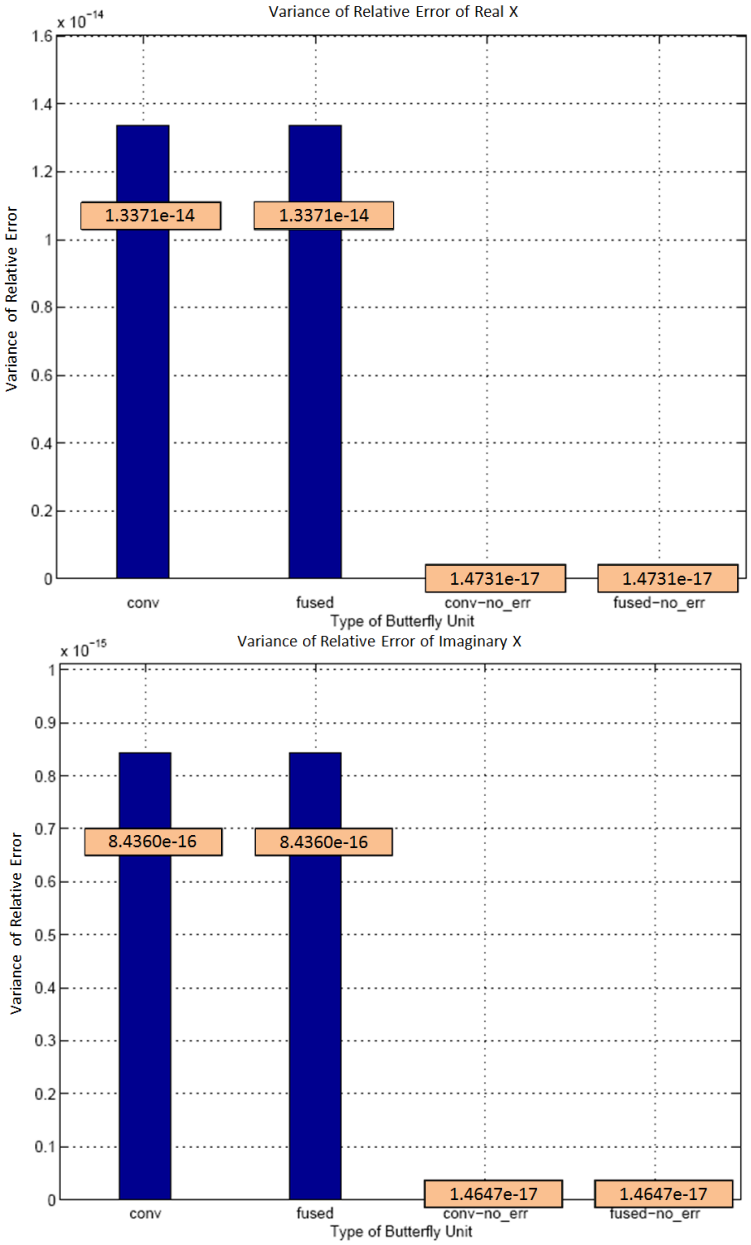
butterfly unit is the same as the output of Fused Add-Subtract (Fused AS). The fused AS has no effect on the accuracy.



no_err means 'without input conversion error'

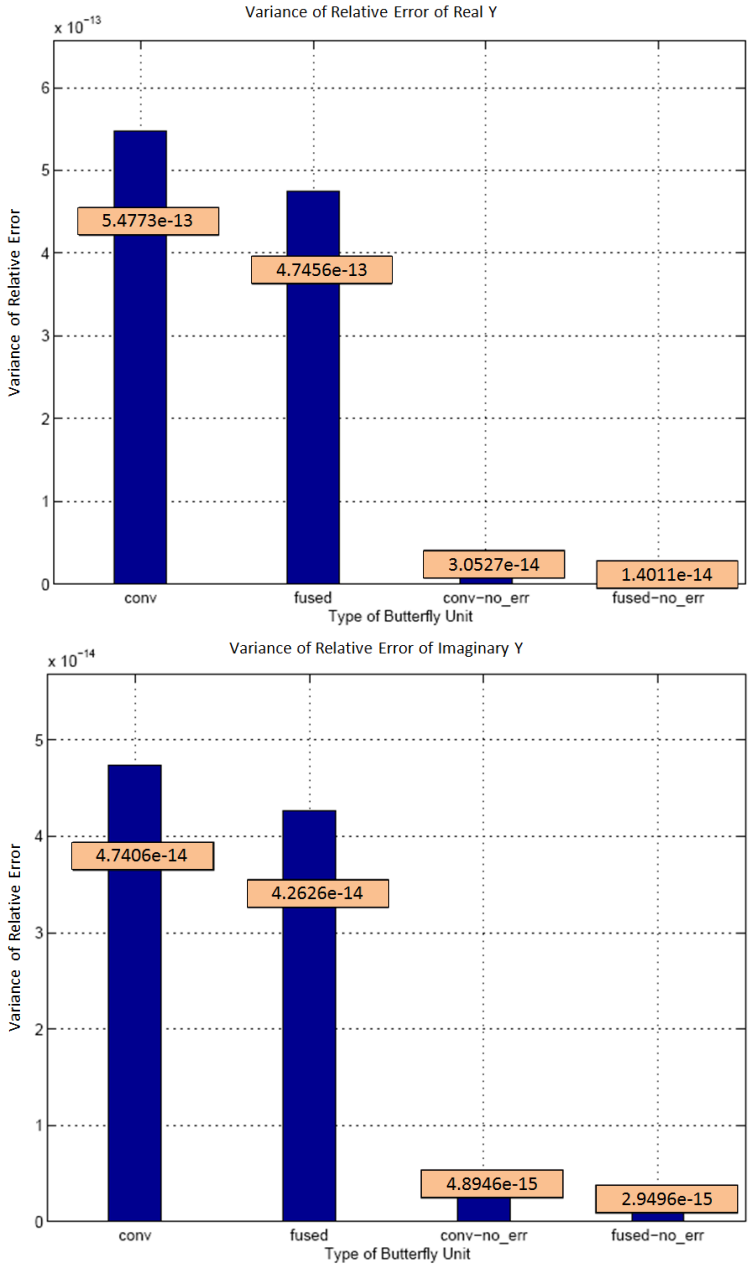
Figure 4.5: Average Relative Absolute Error of Y_{re} and Y_{im}.

The average relative absolute error of Y_{re} and Y_{im} from the fused butterfly unit is smaller than that of the conventional butterfly unit.



no_err means 'without input conversion error'

Figure 4.6: Variance of Relative Error of X_{re} and $X_{im.}$



no_err means 'without input conversion error'

Figure 4.7: Variance of Relative Error of Y_{re} and $Y_{im.}$

Chapter 5 - Butterfly Unit with Merged MCMs

This chapter introduces the Radix-2 butterfly unit for a 1024-point FFT implemented using the conventional Multiple-Constant Multiplier (MCM). The butterfly unit with the MCM has lower precision than a floating-point butterfly unit; however, it consumes lower-power. In addition, a novel butterfly architecture designed by merging neighbor MCMs is presented. The novel butterfly architecture provides two options; it either reduces the relative error or it lowers the power; the average relative absolute error can decrease by 69% without changing the power consumption or the average relative absolute error can be the same with 28.8% lower power.

5.1 Butterfly Unit with MCM

The number of internal shifters of MCM affects the relative error of the MCM output. After making a model of the MCM, the relative error was analyzed as a function of the number of internal shifters. In the analysis, one input of the MCM was a fixed 1024-point coefficient vector and the other input was a random variable. The following plots show the average and variance of the relative error as a function of the number of internal shifters; the number of coefficients of MCM in the plots indicates the number of internal shifters. When the number of shifters increases, the average and variance of the relative error decrease; however, as the number of internal shifters increases, the power consumption increases. MCM has 0% relative error when there are more than eleven internal shifters. MCM with four shifters was selected for this project because it has reasonable relative error and power consumption based on the experiment by Lee [12].

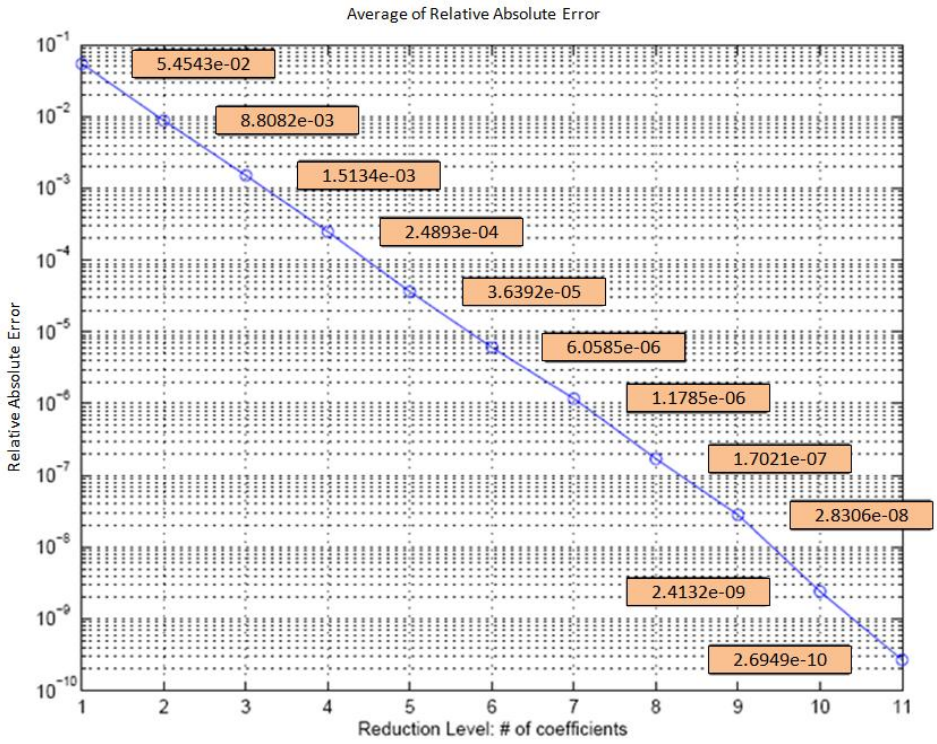


Figure 5.1: Average Relative Absolute Error of MCM.

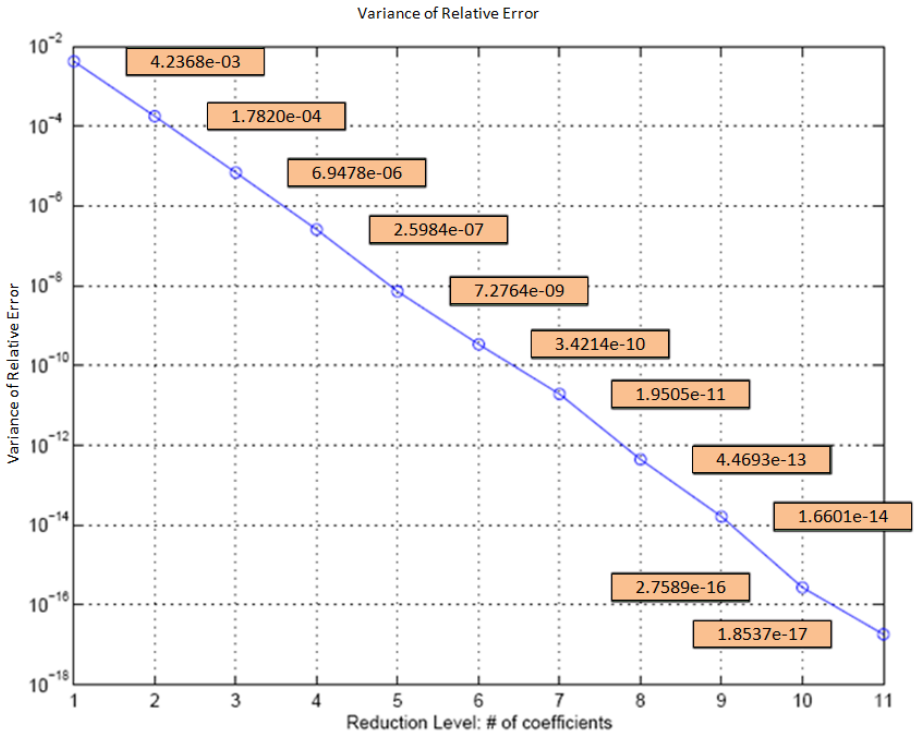


Figure 5.2: Variance of Relative Error of MCM.

Example 5.1 Operation by MCM

1 0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0 1 0 0 0 (Coefficient =*490F88*)

1-1 0 1-1 0 1-10 0 0 1 0 0 0 0-1 0 0 1-1 0 0 0 (Booth-recoded coefficient =*490F88*)

1 0 0 1 0 0 1 0 0 0 1 0 0 0 0-1 0 0 0 1 0 0 0 (Modified booth-recoded coefficient = *490F88*)

Shifted input by 22 + shifted input by 19 + shifted input by 16

+ shifted input by 12 - shifted input by 7 + shifted input by 3 = *490F88* input*

where 22,19,16,12,7,3 : Shift-Coefficients

Example 5.1 shows the operation of the MCM. If four internal shifters are used for the coefficient, *490F88*, the relative absolute error is $2.5062*10^{-5}$ while a MCM with six shifters has 0% relative absolute error. Figure 5.3 shows the block diagram of MCMs for the red circled part of the butterfly unit. Each MCM has four shifters. The shifters are controlled by variables Ci4, Ci3, Ci2, Ci1, Cr4, Cr3, Cr2, and Cr1 that present the amount of shifts (shift-coefficient).

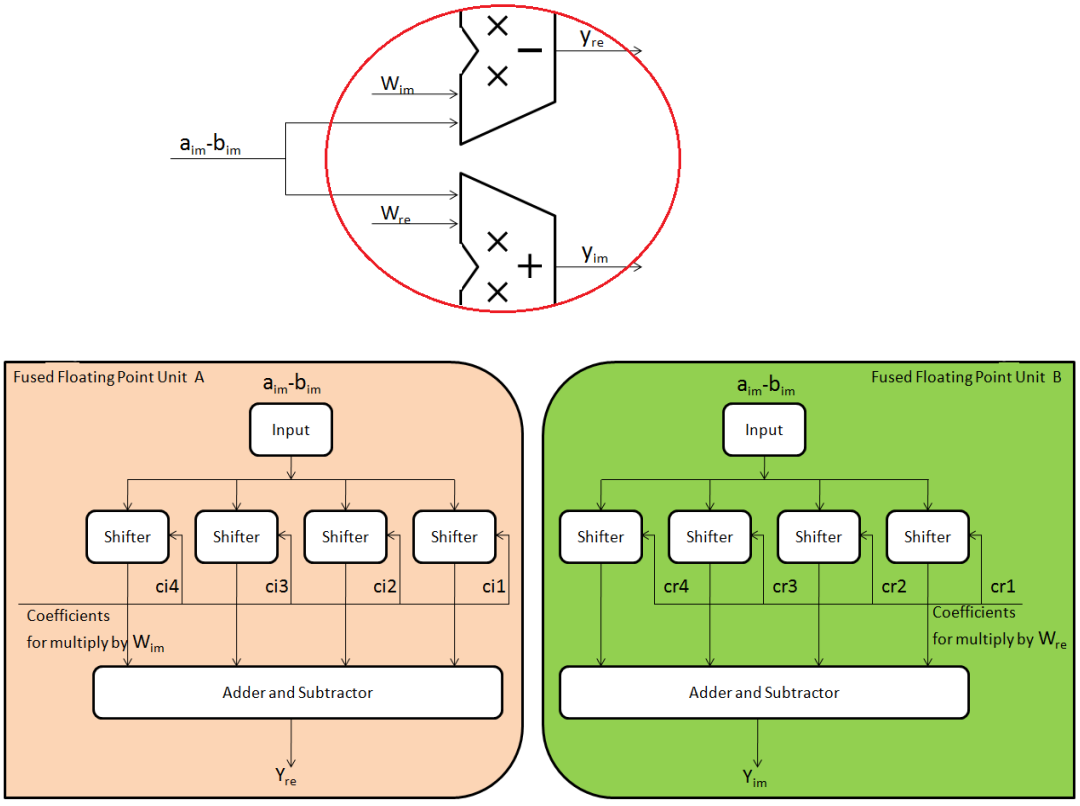


Figure 5.3: Block Diagram of a Conventional Multiple Constant Multiplier in Butterfly Unit.

5.2 New Butterfly Unit implemented by merging Neighbor MCMs

In the butterfly unit using MCMs, some of the shift-coefficients for W_{re} and W_{im} are the same. For example, the shift-coefficients of $W_{re}^{1009}{}_{1024}$ are 22, 19, 16, and 14. The shift-coefficients of $W_{im}^{1009}{}_{1024}$ are 23, 16, 14, and 10. Between those coefficients, two coefficients (16 and 14) are the same. If the two neighbor MCMs take the same input as shown in Figure 5.3, having the same shift-coefficients is useful in two aspects. First of all, as shown in the above example, two redundant shifters can be turned off. This can reduce the power consumption while the relative absolute error remains constant. Second, if the two redundant shifters accept additional shift-coefficients such as 7 for W_{re} and 8 for W_{im} without turning off, the relative absolute error decreases with the same power consumption. The output from accepting another coefficient is the same as the output from MCM with an additional shifter. The following figure shows the percentage of the shift-coefficients that are the same.

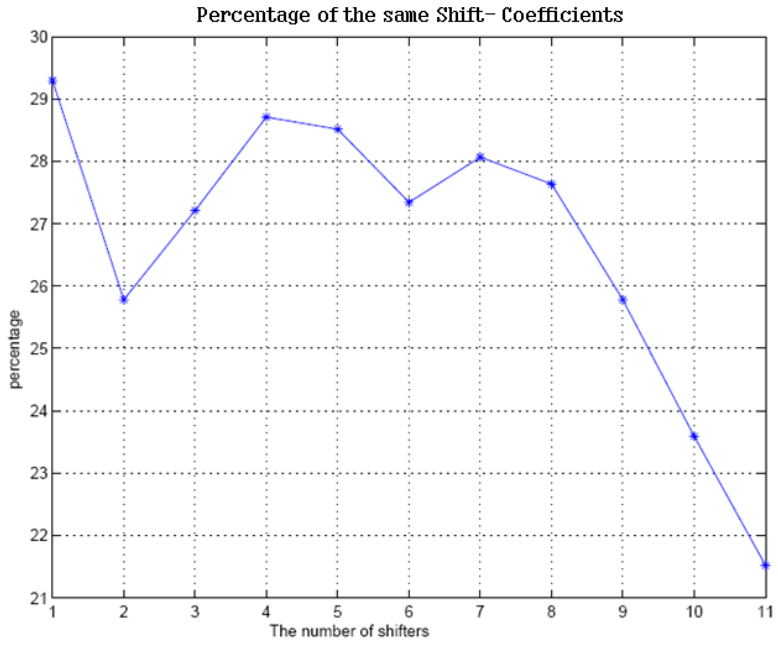


Figure 5.4: Percentage of the Same Shift-Coefficient.

In a four-shifter-based MCM, 28.8% of the shift-coefficients are the same. In other words, 28.8 % of the shifters can be turned off or can take additional shift-coefficients Figure 5.4 shows the merged MCM. Figure 5.5 presents the new butterfly unit architecture with the merged MCM.

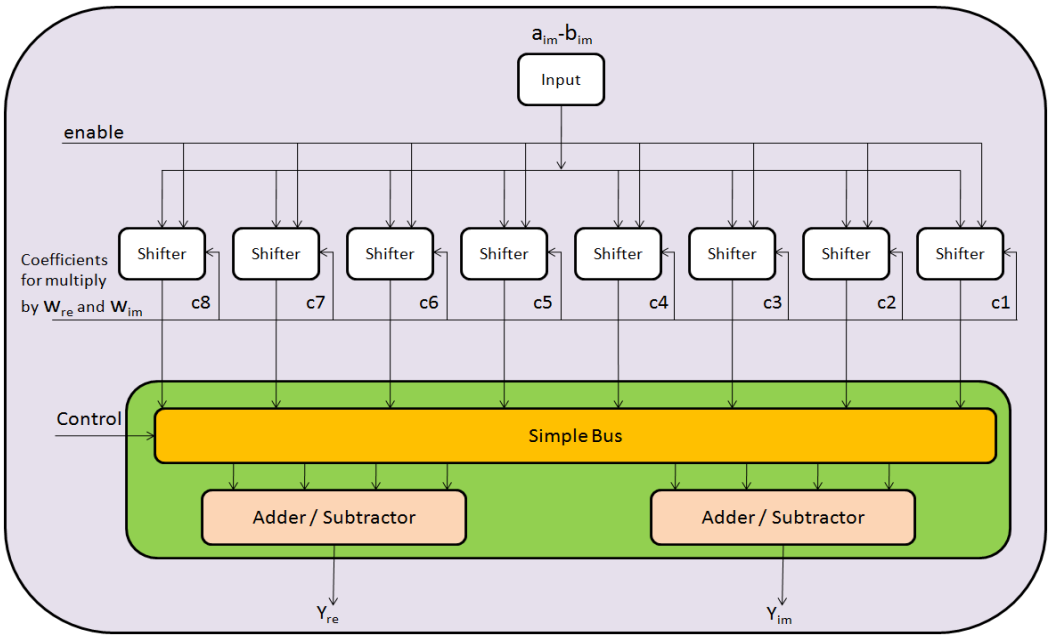


Figure 5.5: Merged MCM.

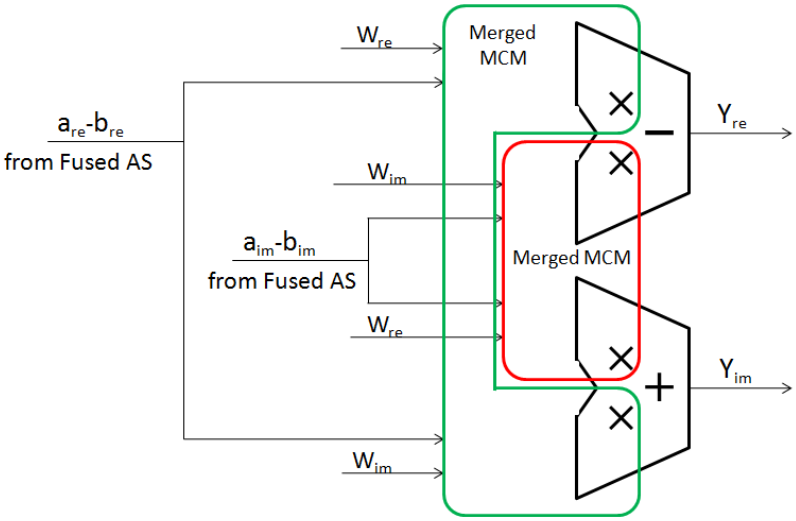


Figure 5.6: Butterfly Unit with Merged MCM.

If the merged MCM takes another shift-coefficient -complete operation mode- instead of the sleep-mode, the relative error decreases. Figure 5.7 shows how the percentage of the average relative absolute error decreases. Figure 5.8 compares the average relative absolute error in three cases where the fused butterfly unit utilizes conventional multipliers, conventional MCMs, or complete operation mode merged MCMs.

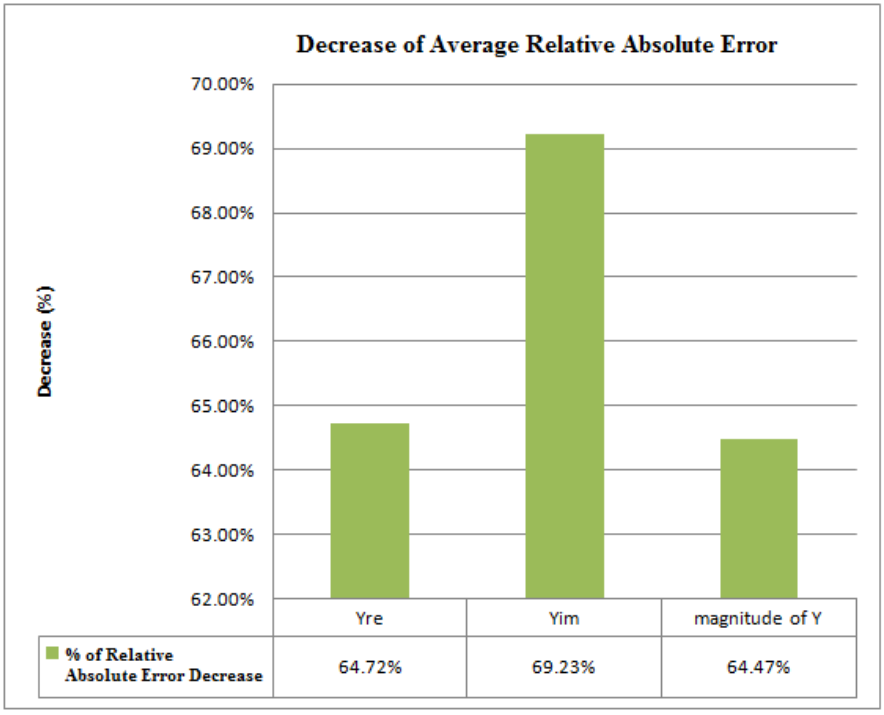


Figure 5.7: Decrease of Average Relative Absolute Error (%).

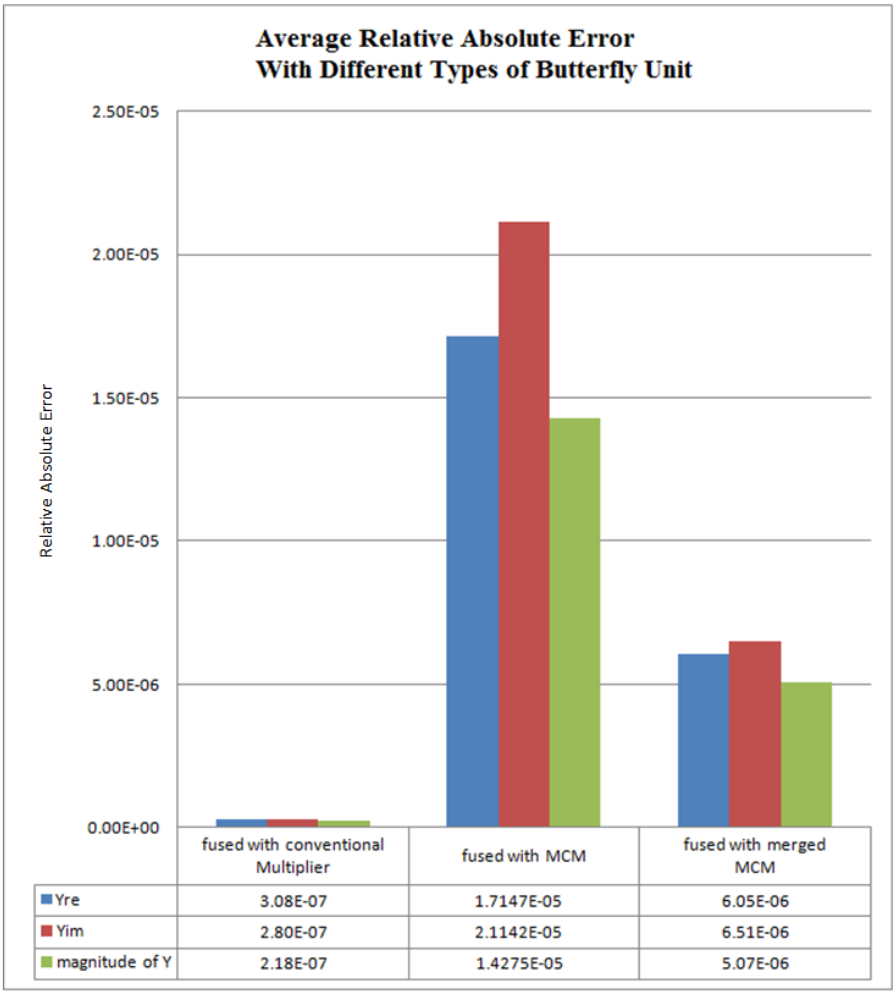


Figure 5.8: Average Relative Absolute Error with Different Types of Butterfly Unit.

Chapter 6 - Conclusion and Future Work

This report examines low-power and high-performance FFT architectures such as FFTs with fuse butterfly arithmetic units. In addition, a butterfly unit with Multiple-Constant Multiplication can be a solution for low-power and high-performance. Furthermore, a new butterfly architecture using merged MCMs was introduced. The merged MCMs can operate in two modes: low-power with modest relative error and low-relative error with modest power. The first operation maximizes power saving. The second operation provides some power saving as well. For example, if a constant relative error is required, the butterfly unit can satisfy the error requirement without increasing the number of internal MCM shifters by using merged MCM; the number of MCM shifters is proportional to power consumption.

In this report, the butterfly unit with merged MCM was designed as a high-level model. Therefore, for the future work, the butterfly unit will be implemented at the gate level and its power will be measured by Hspice and a power estimator such as Synopsys PrimeTime-PX. Moreover, the merged MCM architecture can be improved by tapping the internal values of a logarithmic shifter. For example, if the shift-coefficient is 22 and the MCM shifter is a logarithmic shifter, the internal values of the shifter are 16, 4, and 2. In this case, the shift-coefficient 16, 4 and 2 can be tapped during shifting by 22. The control logic for the improved butterfly should be designed in a way to minimize the power consumption of the control logic. These topics will be further studied in my future work.

Bibliography

[1] *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Standard 754-1985.

[2] M. Ercegovac and T. Lang, *Digital Arithmetic*, San Francisco: Morgan-Kaufmann Publishers, 2006.

[3] Steve Hollasch, *IEEE Standard 754 Floating Point Numbers*, Microsoft Corporation, 2005.

[4] Eric Charles Quinnell, *Floating-Point Fused Multiply-Add Architectures*, Ph.D. Dissertation, University of Texas at Austin, 2007.

[5] Hani Hassan Mustafa Saleh, *Fused Floating-Point Arithmetic For DSP*, Ph.D. Dissertation, University of Texas at Austin, 2009.

[6] R. K. Yu and G. B. Zyner, "167 MHz floating-point multiplier," *Proc. 12th IEEE Symposium on Computer Arithmetic*, pp. 149-154, July 1995.

[7] R. K. Montoye, E. Hokenek and S. L. Runyon, "Design of the IBM RISC System/6000 floating-point execution unit," *IBM Journal of Research & Development*, Vol. 34, pp. 59-70, 1990.

[8] E. Hokenek, R. Montoye and P.W. Cook, "Second-Generation RISC Floating Point with Multiply-Add Fused," *IEEE Journal of Solid-State Circuits*, Vol. 25, pp. 1207-1213, 1990.

[9] W. Han, T. Arslan, A. T. Erdogan and M. Hasan, "Multiplier-less Based Parallel-Pipelined FFT Architecture for Wireless Communication Applications," *Acoustics, Speech, and Signal Processing*, Vol. 5, pp. 45-48, 2005.

[10] H. H. Saleh and E. E. Swartzlander, Jr., "A Floating-point Fused Dot-Product Unit," *IEEE International Conference on Computer Design, ICCD*, pp. 427-431, 2008.

[11] M. Potkonajak, M. Srivastava and A. Chandrakasan, "Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination," *IEEE Trans. on CAD of Integrated Circuits and Systems*, Vol. 15, No. 2, 1996.

[12] Y. Lee, J. Park and K. Chung, "Low Power Constant Multiplier with Variable Precision Computing Capability," *Solid-State and Integrated-Circuit Technology, ICSICT,* pp. 2120-2123, 2008.

[13] Y. Lee, J. Park and K. Chung, "Design of Low Power MAC Operator with Dual Precision Mode," *Embedded and Real-Time Computing Systems and Applications, RTCSA,* pp. 309-318, 2007.

[14] E. E. Swartzlander, Jr. and H. H. Saleh, "Fused floating-point arithmetic for DSP," *Conference Record of the Forty-Second Asilomar Conference on Signals, Systems and Computers,* pp. 767-771, 2008.

[15] H. H. Saleh and E. E. Swartzlander, Jr., "A Floating-point Fused Add-Subtract Unit," *The Fifty-First Midwest Symposium on Circuits and Systems, MWSCAS,* pp. 519-522, 2008.

[16] E. E. Swartzlander, Jr., EE382V *Floating-point Arithmetic and Design Course Notes,* University of Texas at Austin, 2006

VITA

Min, Jae Hong was born in Gwangmyeong, South Korea in 1981. In 2008, he received his Bachelor degree in Electrical Engineering from both the State University of New York at Stony Brook, NY and Ajou University, South Korea based on the dual degree program between the schools. In August 2008, he began his study in the electrical engineering graduate program at The University of Texas at Austin. Under the supervision of Dr. Earl E. Swartzlander Jr., he has worked on Fused Floating-Point Arithmetic. He is especially interested in low-power fused floating-point arithmetic based on data pattern dependency.

Permanent Address: Gwangmyeong 4-dong Gwangmyeong-Si, Gyeonggi-do,
Republic of Korea.

This thesis was typed with Word by the author.